

Redes de Computadores

Prof. Daniel Ludovico Guidoni

guidoni@ufop.edu.br

Redes de Computadores

Capítulo 2: Camada de Aplicação

Aula 1: Princípios de aplicações de rede

Algumas aplicações de rede

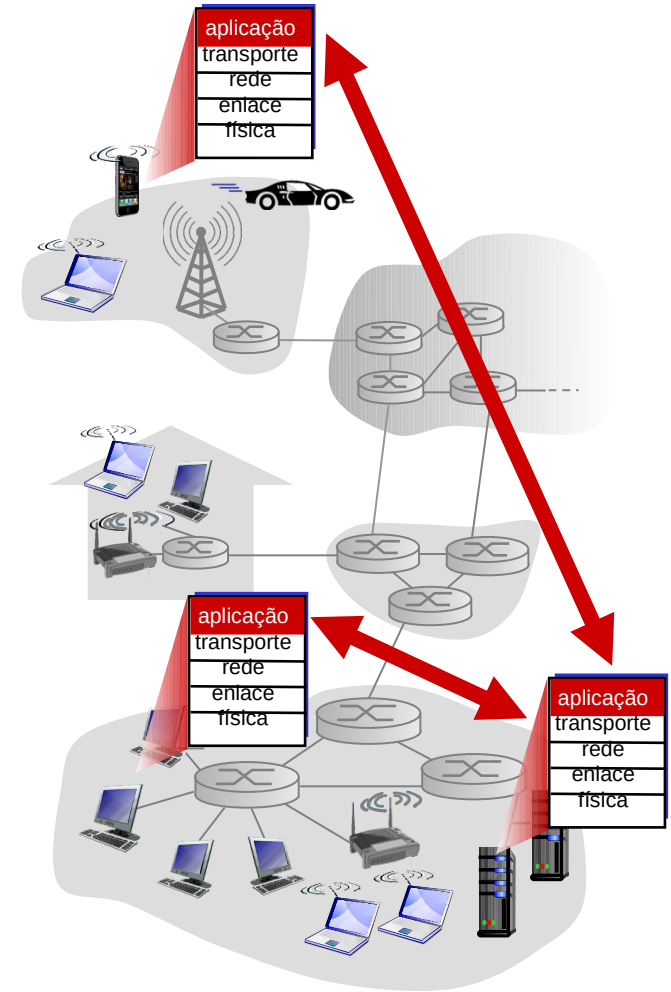
- Correio eletrônico
- A Web
- Mensagens instantâneas
- Login em computador remoto como Telnet e SSH
- Compartilhamento de arquivos P2P
- Jogos multiusuários em rede
- *Streaming* de vídeos armazenados (YouTube, Hulu, Netflix)
- Telefonia por IP (Skype)
- Videoconferência em tempo real
- Busca
- ...



Criando uma aplicação em rede

Programas que

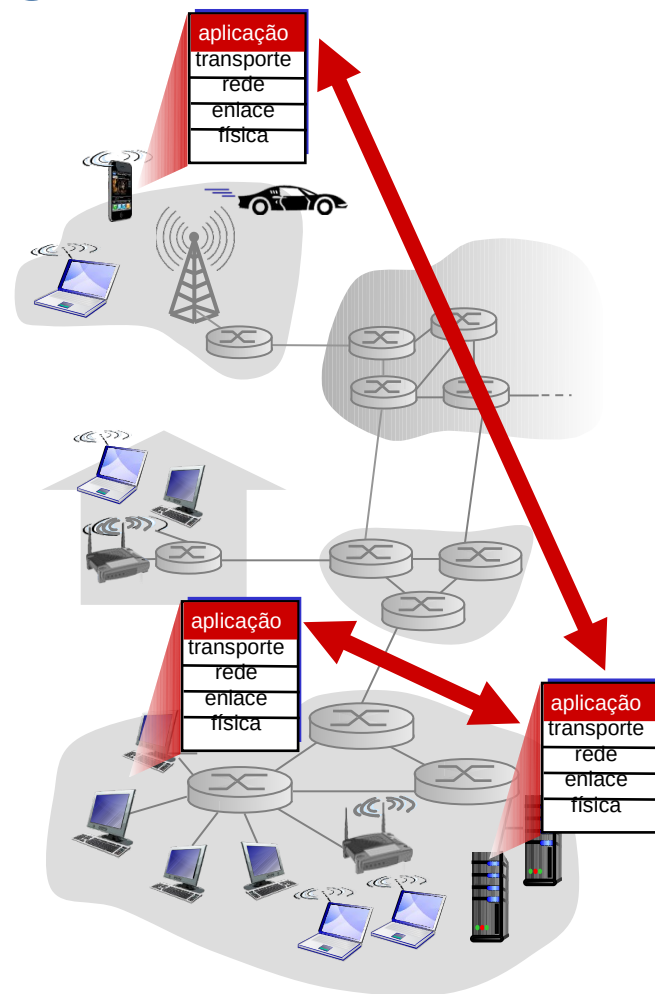
- Executam em (diferentes) sistemas finais
- Comunicam-se através da rede
- p.ex., servidor Web se comunica com o navegador



Criando uma aplicação em rede

Programas não relacionados ao núcleo da rede

- Dispositivos do núcleo da rede não executam aplicações dos usuários
- Aplicações nos sistemas finais permitem rápido desenvolvimento e disseminação



Arquitetura das aplicações

- Estruturas possíveis das aplicações:
 - Cliente-servidor
 - Peer-to-peer (P2P)

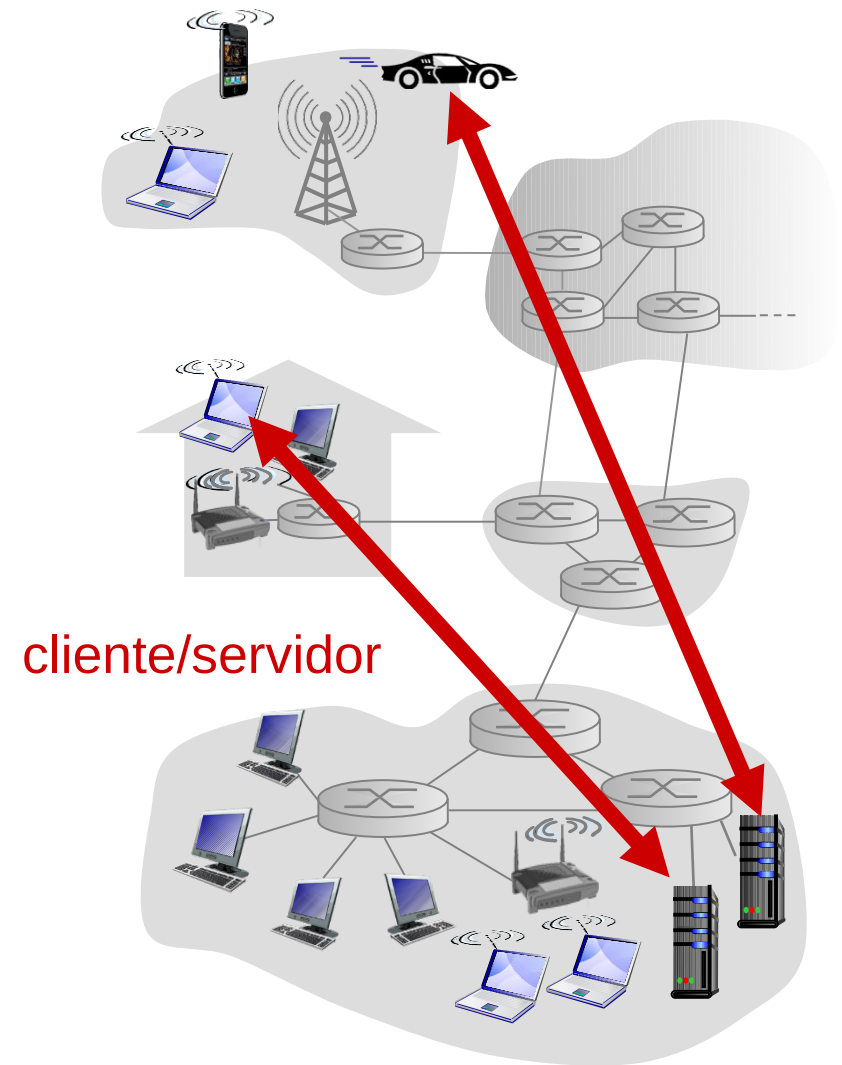
Arquitetura cliente-servidor

Servidor:

- Sempre ligado
- Endereço IP permanente
- Escalabilidade com data centers

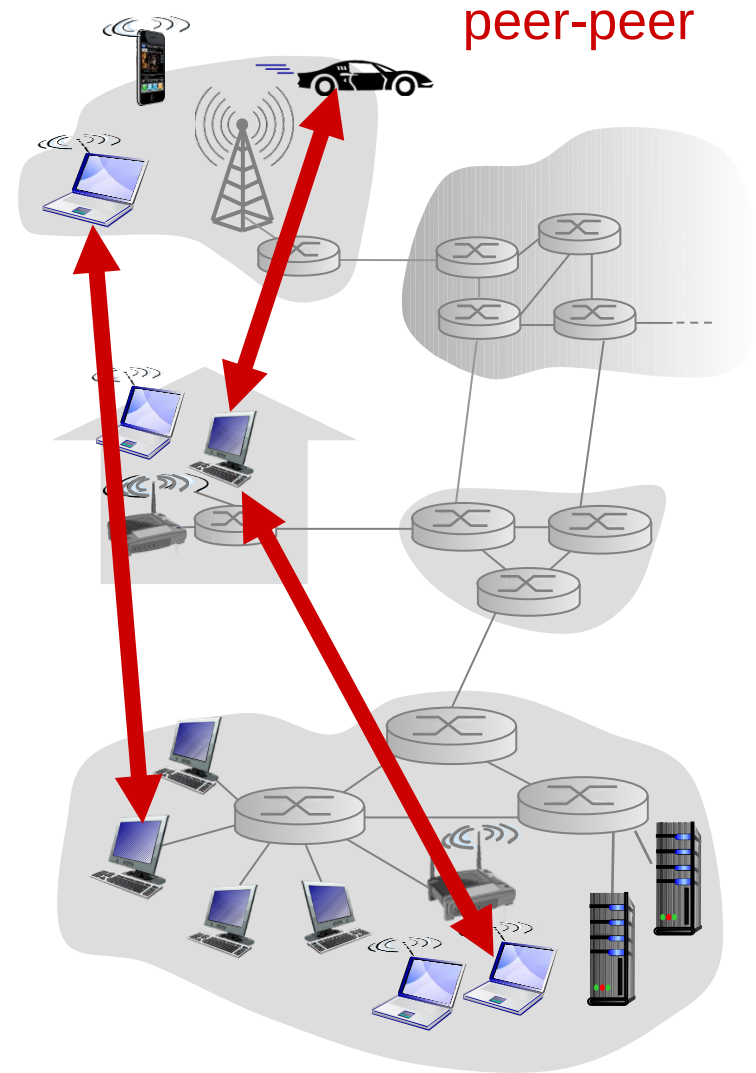
Clientes:

- Comunicam-se com o servidor
- Podem estar conectados intermitentemente
- Podem ter endereços IP dinâmicos
- Não se comunicam diretamente com outros clientes



Arquitetura P2P

- Não há servidor sempre ligado
- **Sistemas finais arbitrários se comunicam diretamente**
- Pares solicitam serviços de outros pares e em troca proveem serviços para outros parceiros:
 - **Autoescalabilidade** – novos pares trazem nova capacidade de serviço assim como novas demandas por serviços.
- Pares estão conectados intermitentemente e mudam endereços IP
 - Gerenciamento complexo

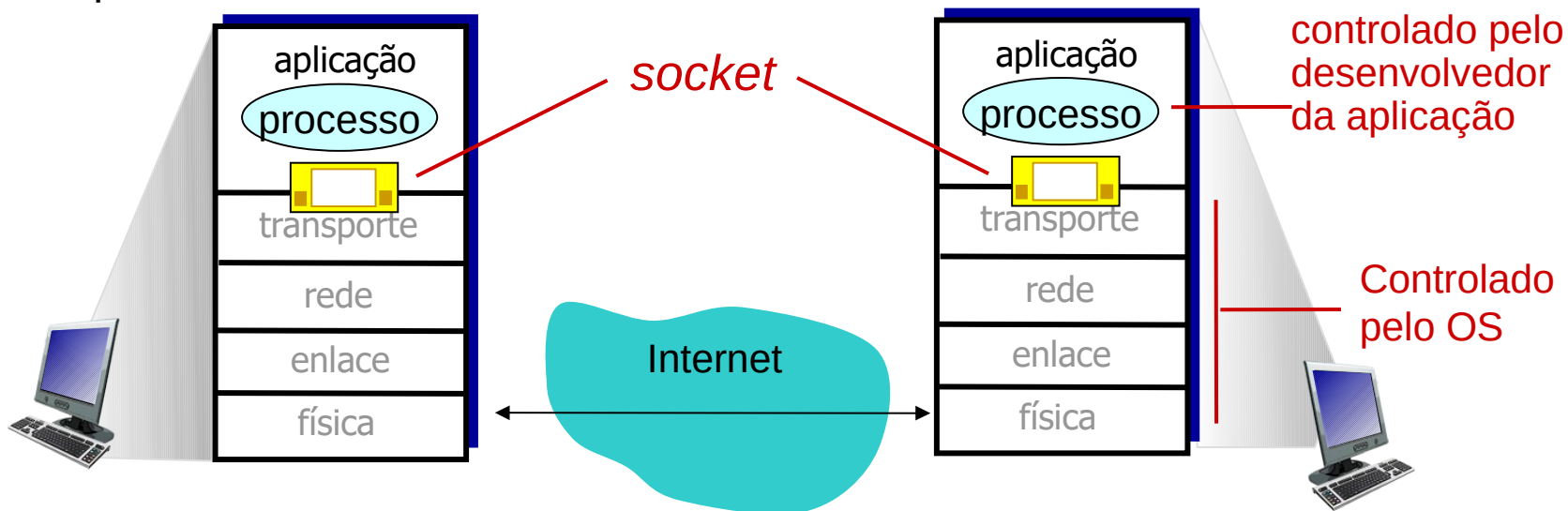


Comunicação entre processos

- **Processo**: programa que executa em um sistema final
 - Processos no mesmo sistema final se comunicam usando **comunicação entre processos** (definido pelo sistema operacional)
 - Processos em sistemas finais distintos se comunicam trocando **mensagens** através da rede
- **Processo Cliente**: processo que inicia a comunicação
- **Processo servidor**: processo que espera ser contatado
- Aplicações em arquiteturas P2P possuem processos clientes e processos servidores

Sockets

- Os processos enviam/ recebem mensagens para/dos seus sockets
- Um socket é análogo a uma porta
 - Processo transmissor envia a mensagem através da porta
 - O processo transmissor assume a existência da infraestrutura de transporte no outro lado da porta que faz com que a mensagem chegue ao socket do processo receptor



Endereçamento de processo

- Para que um **processo** receba mensagens, ele deve possuir um **identificador**
- Cada hospedeiro possui um endereço IP único de 32 bits
- **Pergunta:** o endereço IP do hospedeiro no qual o processo está sendo executado é suficiente para identificar o processo?
- **Resposta:** Não, muitos processos podem estar executando no mesmo hospedeiro
- O identificador inclui tanto o endereço IP quanto os números das portas associadas com o processo no hospedeiro
- Exemplo de números de portas:
 - Servidor HTTP: 80
 - Servidor de Correio: 25
- Para enviar uma msg HTTP para o servidor Web `gaia.cs.umass.edu`
 - Endereço IP: 128.119.245.12
 - Número da porta: 80
- Mais sobre isto posteriormente.

Protocolos da camada de aplicação definem

- **Tipos de mensagens trocadas**
 - Ex: mensagens de requisição e resposta
- **Sintaxe das mensagens**
 - Campos presentes nas mensagens e como são identificados
- **Semântica das mensagens**
 - Significado das informações nos campos
- **Regra** para quando os processos enviam e respondem às mensagens
- **Protocolos abertos**
 - Definidos em RFCs
 - Permitem a Interoperação
 - Ex: HTTP, SMTP, DNS....

Quais serviços uma aplicação precisa?

Integridade dos dados

- algumas aplicações (p.ex., transf. de arquivos, transações web) requerem uma transferência 100% confiável
- outras (p.ex. áudio) podem tolerar algumas perdas

Temporização

- algumas apps (p.ex., telefonia Internet, jogos interativos) requerem baixo retardo para serem “viáveis”

Vazão

- algumas apps (p.ex., multimídia) requerem quantia mínima de vazão para serem “viáveis”
- outras apps (“aplicações elásticas”) conseguem usar qq quantia de banda disponível

Segurança

- Criptografia, integridade dos dados, ...

Requisitos de aplicações em redes

Aplicação	Perda de dados	Vazão	Sensibilidade ao tempo
Transferência / download de arquivo	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns kbits/s)	Não
Telefonia via Internet/ videoconferência	Tolerante à perda	Áudio: alguns kbits/s – 1Mbit/s Vídeo: 10 kbits/s – 5 Mbits/s	Sim: décimos de segundo
Áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Poucos kbits/s – 10 kbits/s	Sim: décimos de segundo
Mensagem instantânea	Sem perda	Elástico	Sim e não

Serviços providos pelos protocolos de transporte da Internet

Serviço TCP

- transporte confiável entre processos remetente e receptor
- controle de fluxo: remetente não vai “afogar” receptor
- controle de congestionamento: estrangular remetente quando a rede estiver carregada
- não provê: garantias temporais ou de banda mínima
- orientado a conexão: apresentação requerida entre cliente e servidor

Serviços providos pelos protocolos de transporte da Internet

Serviço UDP

- Transferência de dados não confiável entre processos remetente e receptor
- Não provê: estabelecimento da conexão, confiabilidade, controle de fluxo, controle de congestionamento, garantias temporais ou de banda mínima
- Pergunta: Qual é o interesse em ter um protocolo como o UDP?

Aplicações na internet: protocolos de aplicação e transporte

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP [RFC 5321]	TCP
Acesso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de arquivos	FTP [RFC 959]	TCP
Multimídia em fluxo contínuo	HTTP (por exemplo, YouTube)	TCP
Telefonia por Internet	SIP [RFC 3261], RTP [RFC 3550] ou proprietária (por exemplo, Skype)	UDP ou TCP

Aplicações na internet: protocolos de aplicação e transporte

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP [RFC 5321]	TCP
Acesso a terminal remoto	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferência de arquivos	FTP [RFC 959]	TCP
Multimídia em fluxo contínuo	HTTP (por exemplo, YouTube)	TCP
Telefonia por Internet	SIP [RFC 3261], RTP [RFC 3550] ou proprietária (por exemplo, Skype)	UDP ou TCP

Tornando o TCP seguro

TCP & UDP

- Sem criptografia
- Senhas em texto aberto enviadas aos sockets atravessam a Internet em texto aberto

SSL

- Provê conexão TCP criptografada
- Integridade dos dados
- Autenticação do ponto terminal

SSL está na camada de aplicação

- Aplicações usam bibliotecas SSL que “falam” com o TCP

API do socket SSL

- Senhas em texto aberto enviadas ao socket atravessam a rede criptografadas

A Web e o HTTP

Primeiro uma revisão...

- Páginas Web consistem de objetos
- um objeto pode ser um arquivo HTML, uma imagem JPEG, um applet Java, um arquivo de áudio,...
- Páginas Web consistem de um arquivo base HTML que inclui vários objetos referenciados
- Cada objeto é endereçável por uma URL
- Exemplo de URL:
 - www.ufsj.edu.br

A Web e o HTTP

- Exemplo de URL:

`https://ccomp.ufsj.edu.br/images/documentos/Ementario.pdf`



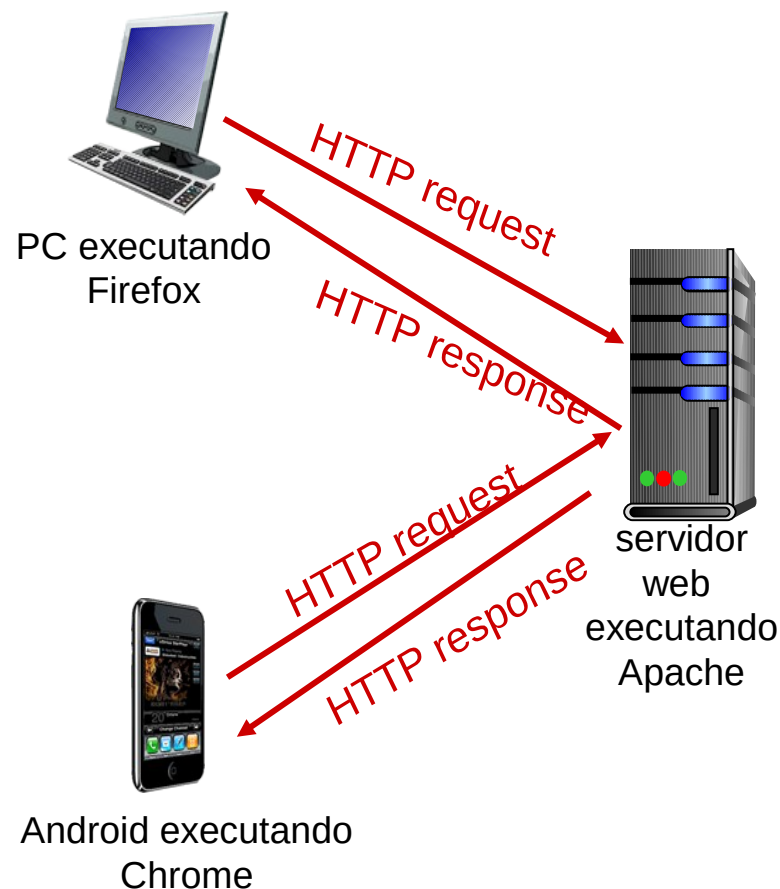
nome do
hospedeiro

nome do caminho

Visão geral do HTTP

HTTP: hypertext transfer protocol

- protocolo da camada de aplicação da Web
- modelo cliente/servidor
- cliente: browser que pede, recebe (usando o protocolo HTTP) e “visualiza” objetos Web
- servidor: servidor Web envia (usando o protocolo HTTP) objetos em resposta a pedidos



Visão geral do HTTP

- Usa serviço de **transporte TCP**:
 - cliente inicia conexão TCP (cria socket) ao servidor, porta 80
 - servidor aceita conexão TCP do cliente
 - mensagens HTTP (mensagens do protocolo da camada de aplicação) trocadas entre browser (cliente HTTP) e servidor Web (servidor HTTP)
 - encerra conexão TCP
- HTTP é “**sem estado**”
 - servidor não mantém informação sobre pedidos anteriores do
- Protocolos que mantêm estados são **complexos**

Conexões HTTP

HTTP não persistente

- No máximo um objeto é enviado numa conexão TCP
 - A conexão é então encerrada
- Baixar múltiplos objetos requer o uso de múltiplas conexões

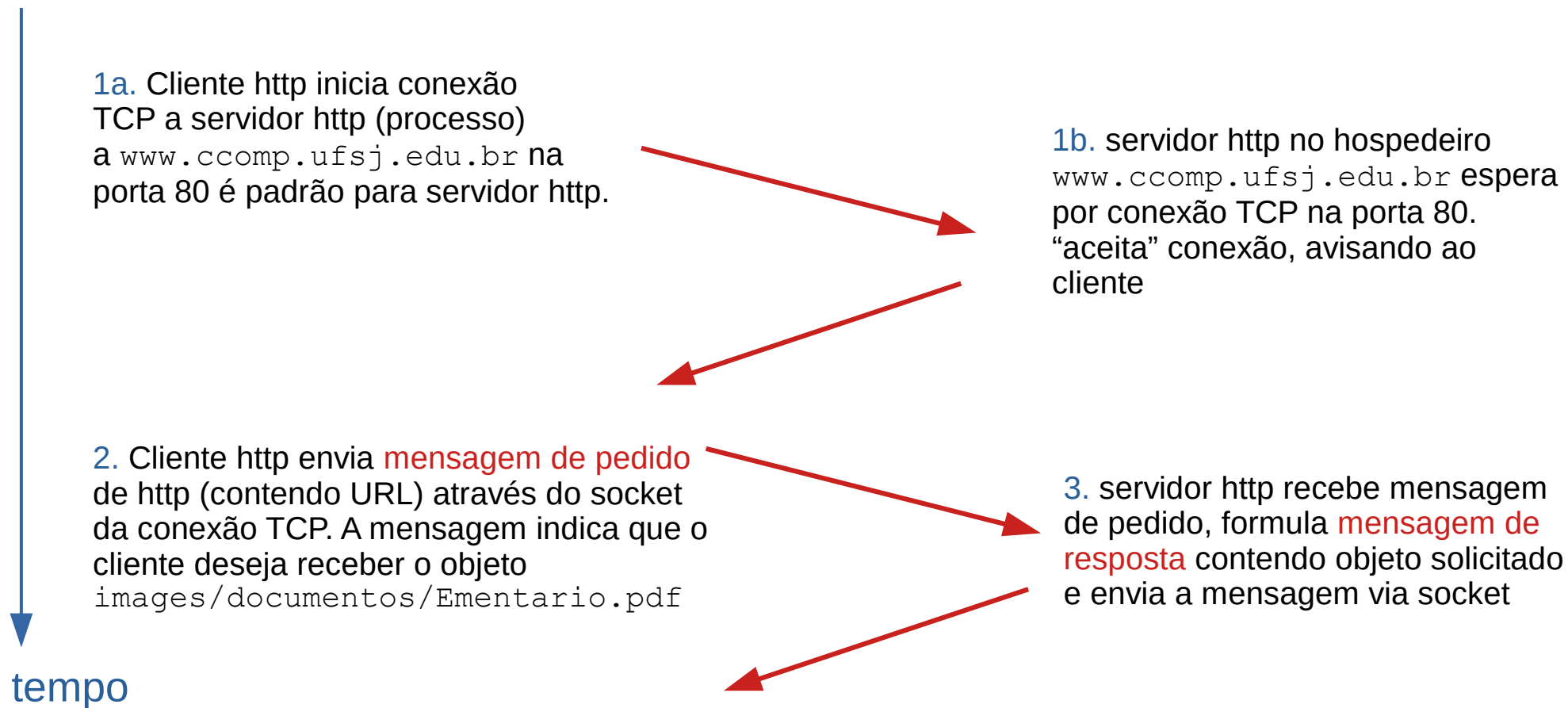
HTTP persistente

- Múltiplos objetos podem ser enviados sobre uma única conexão TCP entre cliente e servidor

Exemplo de HTTP não persistente

Supomos que usuário digita a URL:

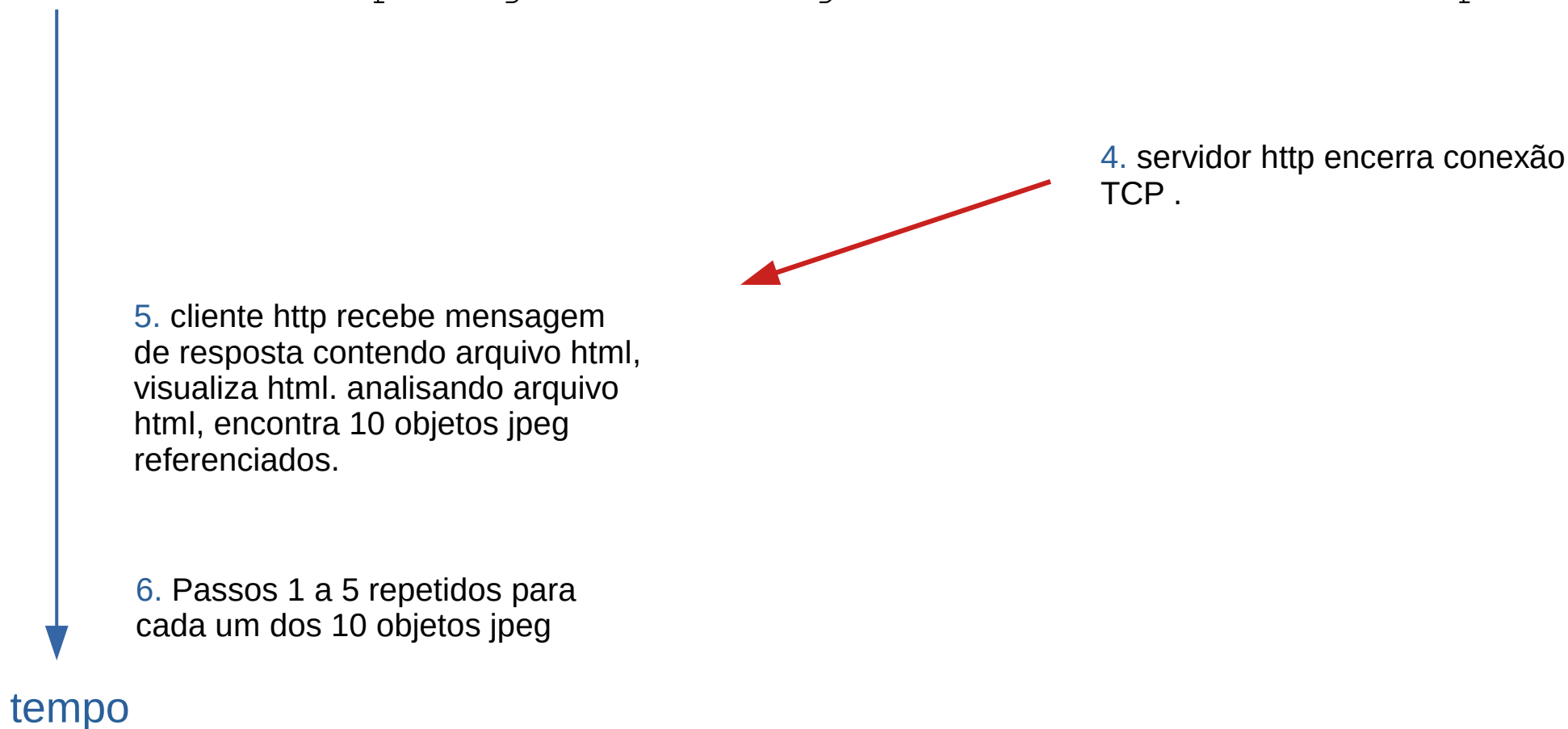
`www.ccomp.ufsj.edu.br/images/documentos/Ementario.pdf`



Exemplo de HTTP não persistente

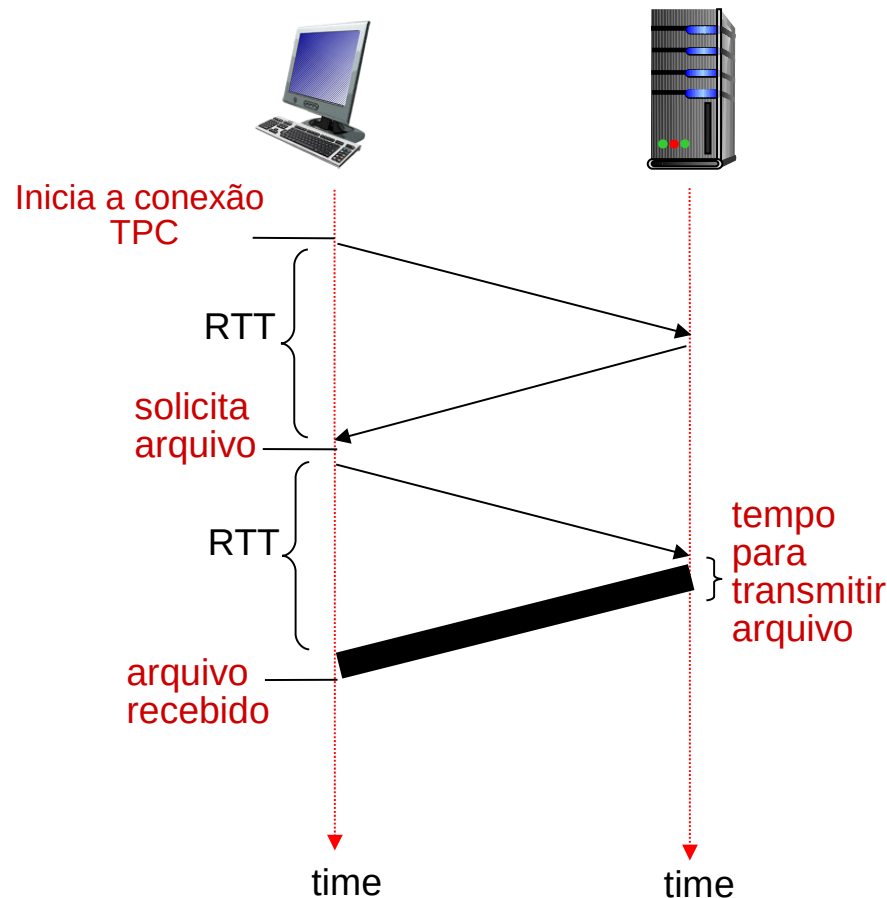
Supomos que usuário digita a URL:

`www.ccomp.ufsj.edu.br/images/documentos/Ementario.pdf`



HTTP não persistente: tempo de resposta

- Definição de RTT (Round Trip Time)
 - intervalo de tempo entre a ida e a volta de um pequeno pacote entre um cliente e um servidor
- Tempo de resposta:
 - um RTT para iniciar a conexão TCP
 - um RTT para o pedido HTTP e o retorno dos primeiros bytes da resposta HTTP
 - tempo de transmissão do arquivo
- total = $2RTT + \text{tempo de transmissão do arquivo}$



Problemas com o HTTP não persistente

- Requer 2 RTTs para cada objeto
- SO aloca recursos do hospedeiro (overhead) para cada conexão TCP
- Os browser frequentemente abrem conexões TCP paralelas para recuperar os objetos referenciados

HTTP persistente

- O servidor deixa a conexão aberta após enviar a resposta
- Mensagens HTTP seguintes entre o mesmo cliente/servidor são enviadas nesta conexão aberta
- O cliente envia os pedidos logo que encontra um objeto referenciado
- Pode ser necessário apenas um RTT para todos os objetos referenciados

Mensagem de requisição HTTP

- Dois tipos de mensagem HTTP: requisição , resposta
- Mensagem de requisição HTTP:
 - ASCII (formato legível por pessoas)

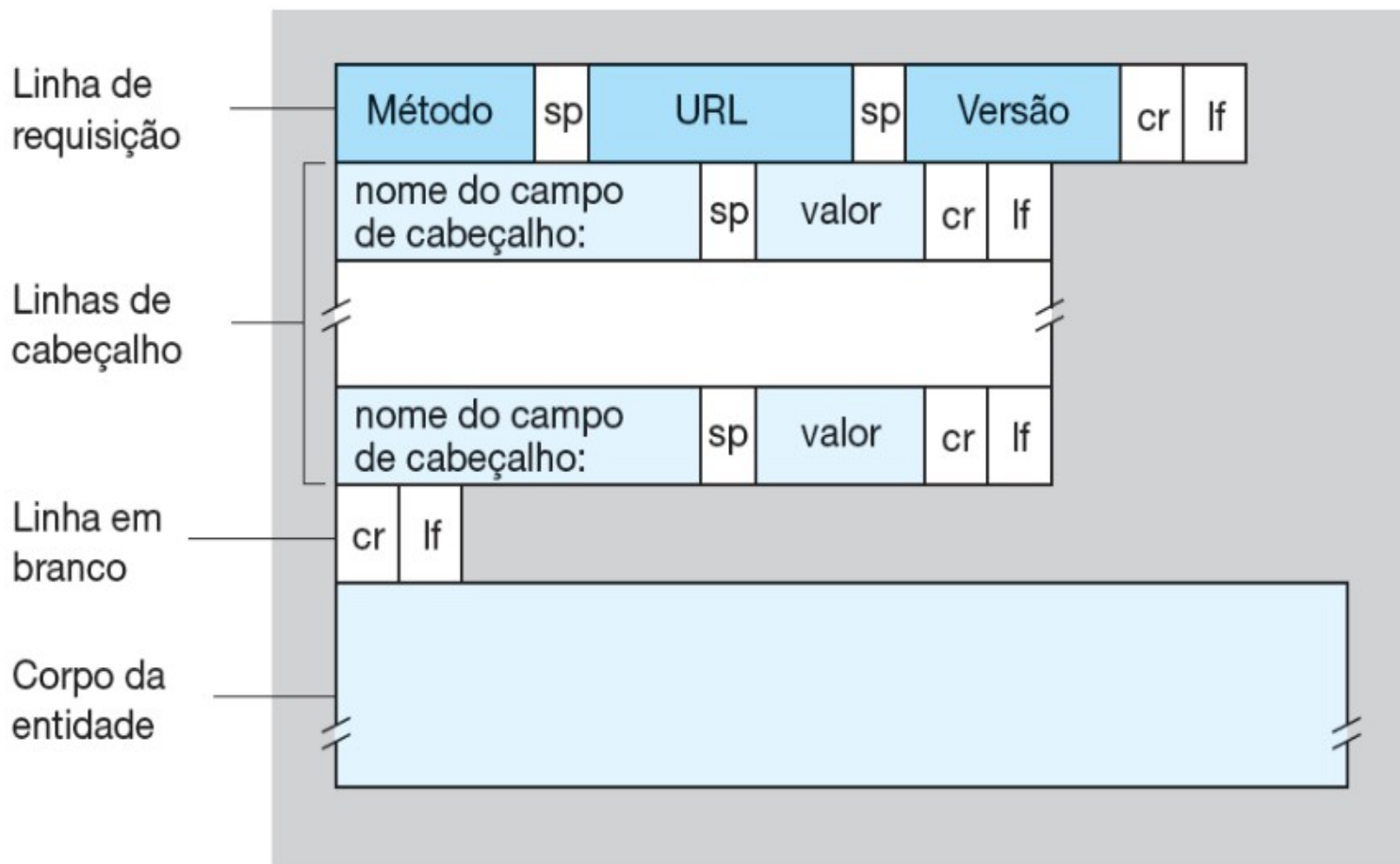
linha de requisição
(comandos GET,
POST, HEAD)

linhas do cabeçalho

Carriage return,
Line feed
Indicam fim de
mensagem

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Mensagem de requisição HTTP: formato geral

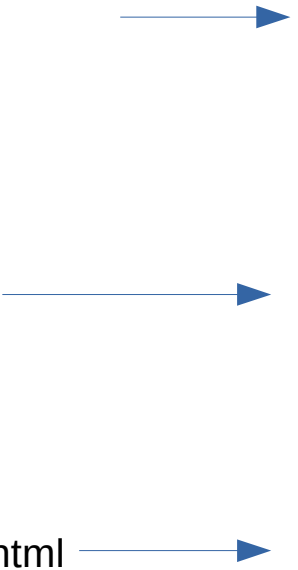


Mensagem de resposta HTTP

Linha de status
(protocolo, código de status
frase de estatus)

linhas de
cabeçalhos

dados: arquivo html
solicitado



```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```


Códigos de status da resposta HTTP

- Na primeira linha da mensagem de resposta servidor->cliente. Alguns códigos típicos:
- **200 OK**
 - sucesso, objeto pedido segue mais adiante nesta mensagem
- **301 Moved Permanently**
 - objeto pedido mudou de lugar, nova localização especificado mais adiante nesta mensagem (Location:)
- **400 Bad Request**
 - mensagem de pedido não entendida pelo servidor
- **404 Not Found**
 - documento pedido não se encontra neste servidor
- **505 HTTP Version Not Supported**
 - versão de http do pedido não usada por este servidor

Cookies: manutenção do “estado da conexão”

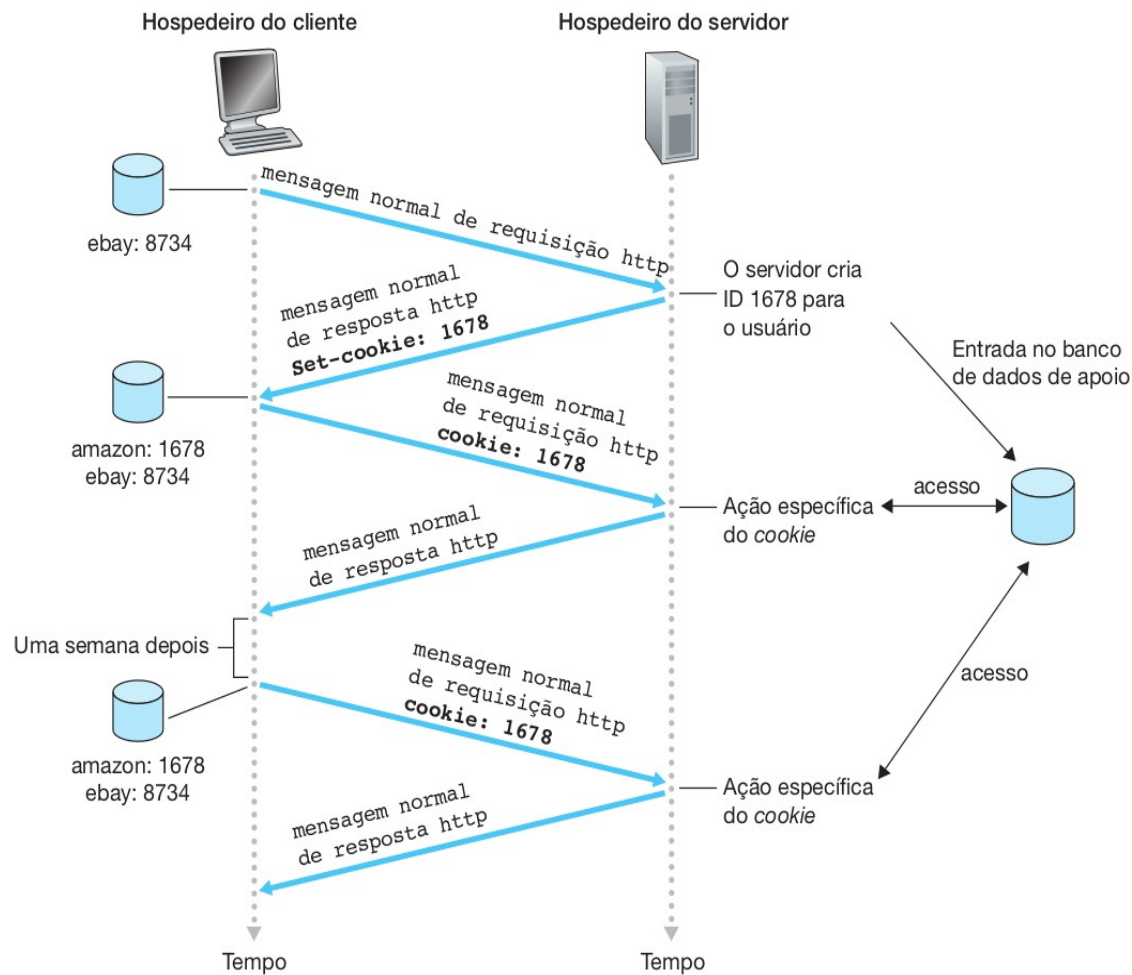
- Principais sites da Web usam *cookies*
- Quatro componentes:
 - 1) linha de cabeçalho do cookie na mensagem de resposta HTTP
 - 2) linha de cabeçalho do cookie na mensagem de pedido HTTP
 - 3) arquivo do cookie mantido no host do usuário e gerenciado pelo browser do usuário
 - 4) BD de retaguarda no sítio Web

Cookies: manutenção do “estado da conexão”

- Exemplo:
 - Daniel Guidoni acessa a Internet sempre do mesmo PC
 - Ele visita um site específico de comércio eletrônico pela primeira vez (Amazon)
 - Quando os pedidos iniciais HTTP chegam no site, o site cria
 - uma ID única
 - uma entrada para a ID no BD de retaguarda



Cookies: manutenção do “estado da conexão”



Cookies: manutenção do “estado da conexão”

O que os cookies podem obter

- Autorização
- Carrinho de compras
- Recomendações
- Estado de sessão do usuário (webmail)

Como manter o “estado”

- Pontos finais do protocolo: mantêm o estado no transmissor/receptor para múltiplas transações
- Cookies: mensagens http transportam o estado

Cookies: manutenção do “estado da conexão”

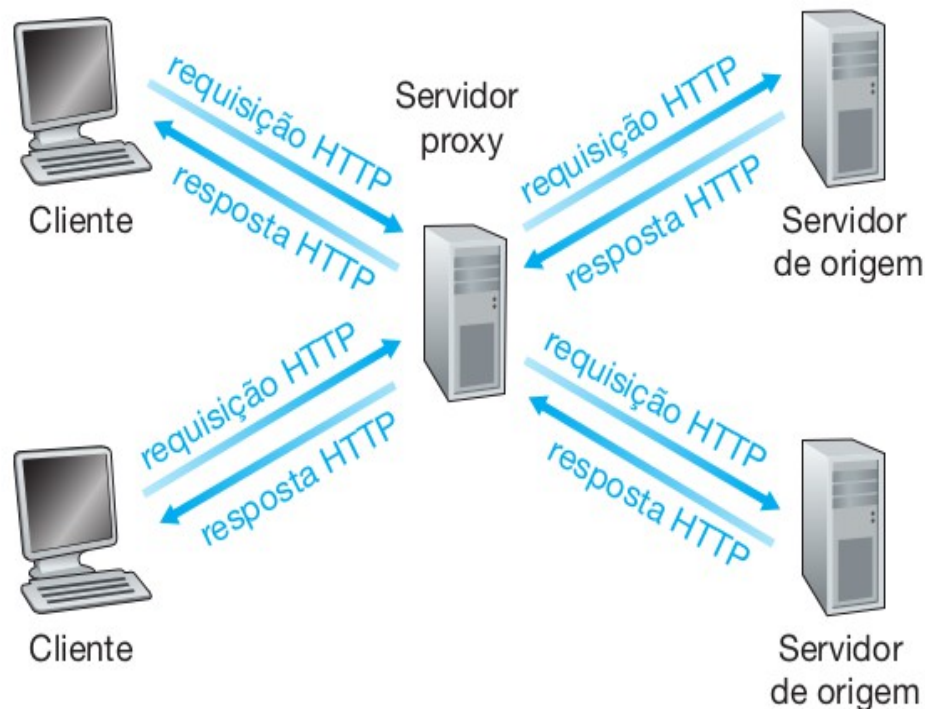
Cookies e privacidade

- Cookies permitem que os sites aprendam muito sobre você
- Você pode fornecer nome e e-mail para os sites

Cache Web (servidor proxy)

Meta: atender pedido do cliente sem envolver servidor de origem

- usuário configura browser : acessos Web via proxy
- cliente envia todos pedidos HTTP ao proxy
 - se objeto estiver no cache do proxy , este o devolve imediatamente na resposta HTTP
 - senão, solicita objeto do servidor de origem, depois devolve resposta HTTP ao cliente



Caches Web

- Para que fazer cache Web?
 - Redução do tempo de resposta para os pedidos do cliente
 - Redução do tráfego no canal de acesso de uma instituição
 - A Internet cheia de caches permitem que provedores de conteúdo “pobres” efetivamente forneçam conteúdo (mas o compartilhamento de arquivos P2P também!)
- Cache atua tanto como cliente quanto como servidor
- Tipicamente o cache é instalado por um ISP (universidade, empresa, ISP residencial)

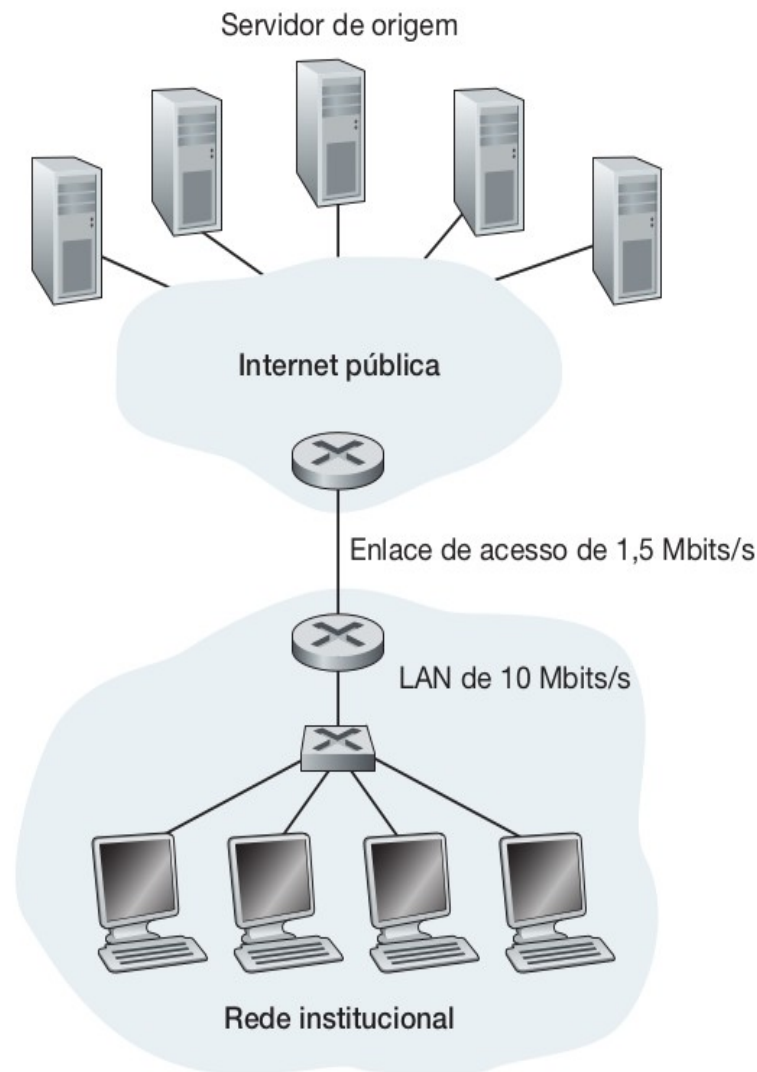
Cache Web: exemplo

Hipóteses

- Tamanho médio de um objeto = 100.000 bits
- Taxa média de solicitações dos browsers de uma instituição para os servidores originais = 15/seg
- Atraso do roteador institucional para qualquer servidor origem e de volta ao roteador = 2seg

Consequências

- Utilização da LAN = 0,15%
- Utilização do canal de acesso = 99% **problema!**
- Atraso total = atraso da Internet + atraso de acesso + atraso na LAN = 2 seg + minutos + microssegundos



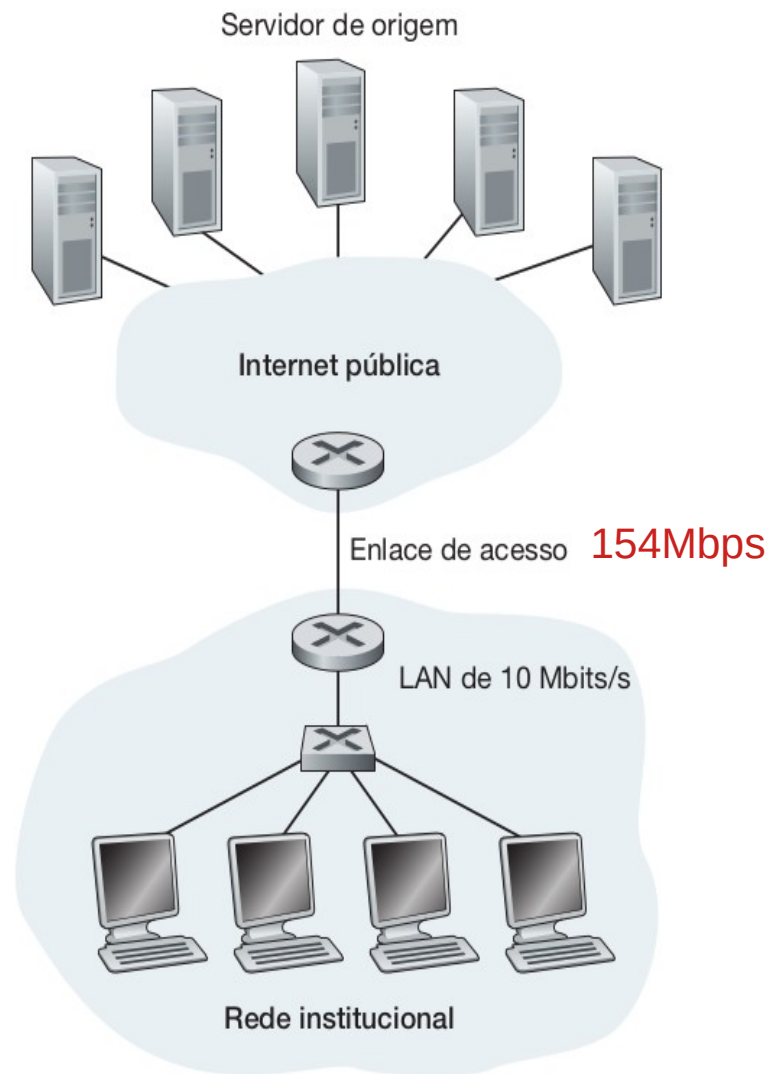
Cache Web: exemplo

Solução em potencial

- Aumento da largura de banda do canal de acesso para, por exemplo, **154 Mbps**

Consequências

- Utilização da LAN = 0,15%
- Utilização do canal de acesso = **9,9%**
- Atraso total = atraso da Internet + atraso de acesso + atraso na LAN = 2 seg + msecs + microssegundos
- Frequentemente este é uma ampliação custosa



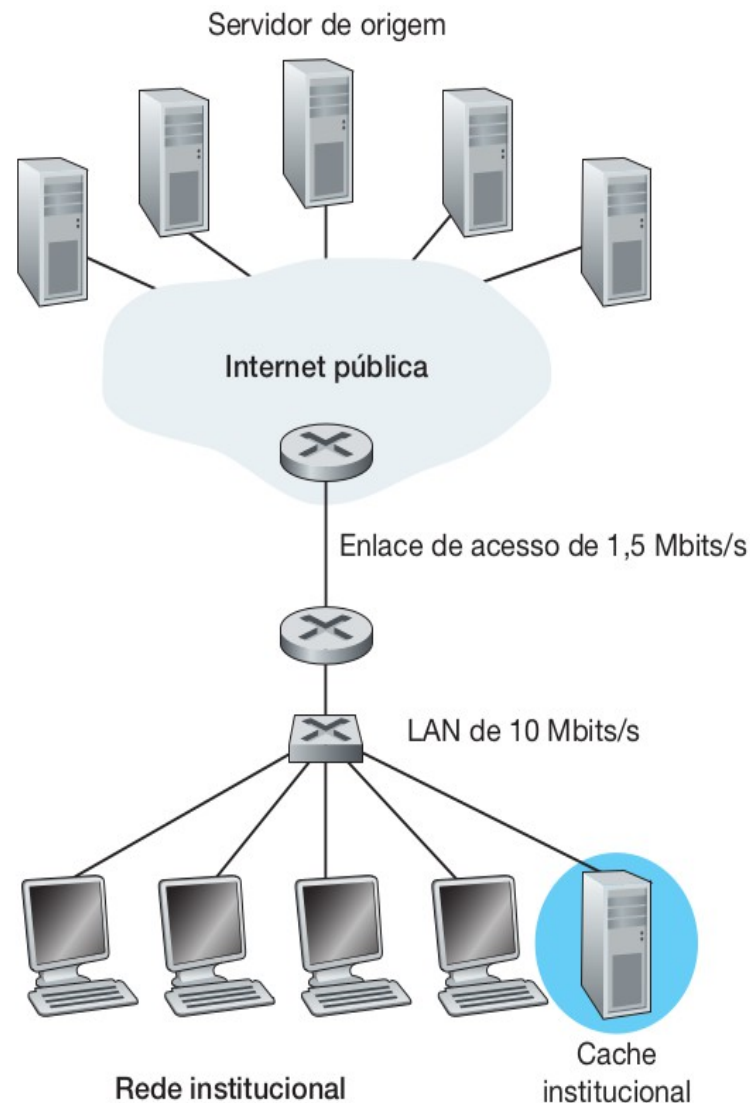
Cache Web: exemplo

Instale uma cache

- Assuma que a taxa de acerto seja de 0,4

Consequências

- 40% dos pedidos serão atendidos quase que imediatamente
- 60% dos pedidos serão servidos pelos servidores de origem
- Utilização do canal de acesso é reduzido para 60%**, resultando em atrasos desprezíveis (ex. 10 mseg)
- Atraso total = atraso da Internet + atraso de acesso + atraso na LAN = $0,6 \cdot 2 \text{ seg} + 0,6 \cdot 0,01 \text{ segs} + \text{msegs} \sim 1,2 \text{ segs}$



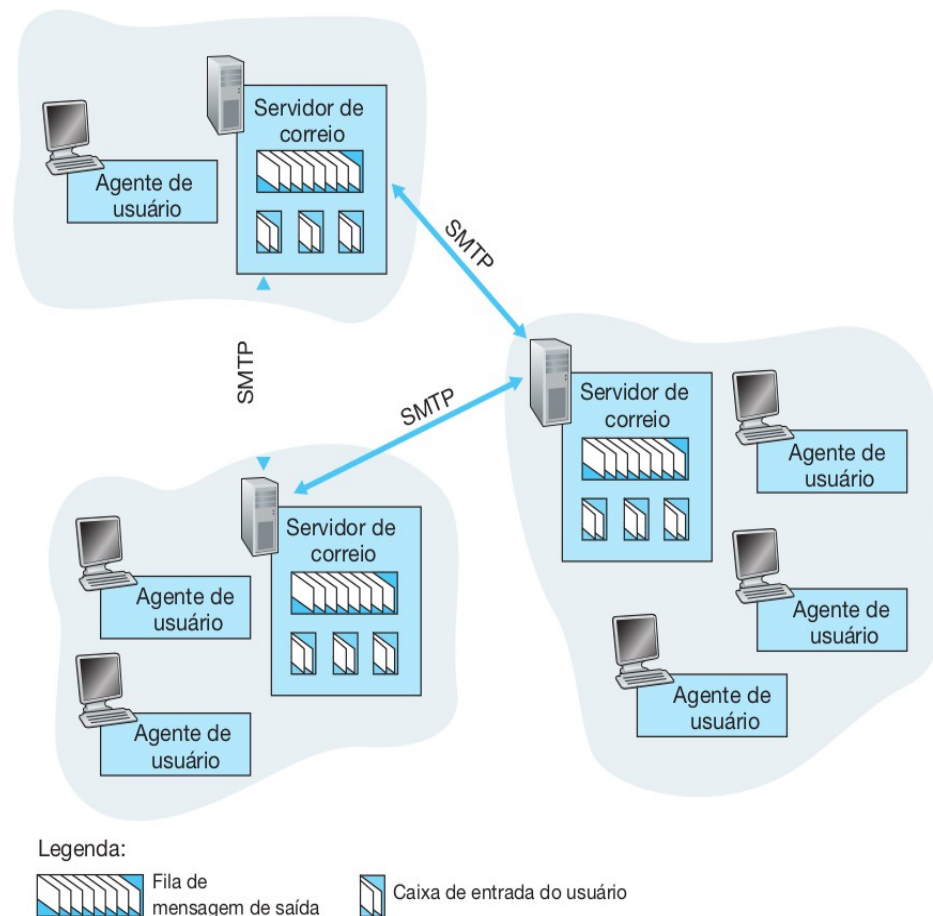
Correio eletrônico

Três grandes componentes:

- agentes de usuário (AU)
- servidores de correio
- Simple Mail Transfer Protocol (SMTP)

Agente de Usuário

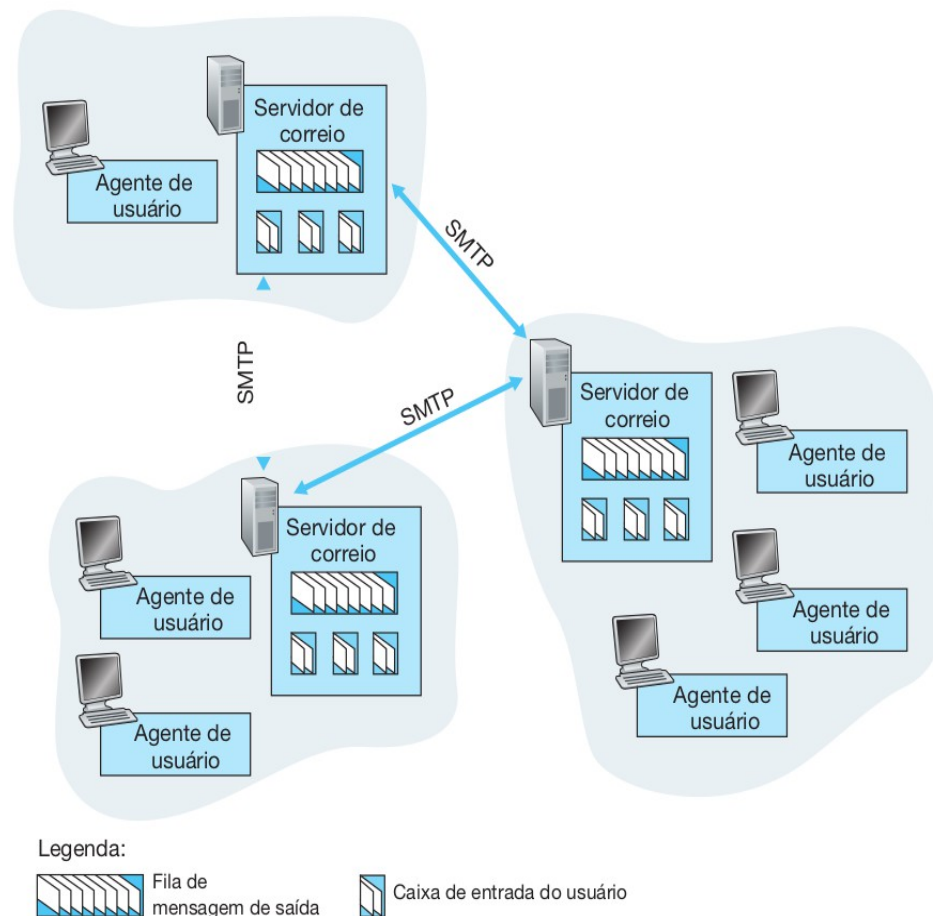
- “leitor de correio”
- compor, editar, ler mensagens de correio
- p.ex., Outlook, Thunderbird, cliente de mail do iPhone
- mensagens de saída e chegando são armazenadas no servidor



Correio eletrônico

Servidores de correio

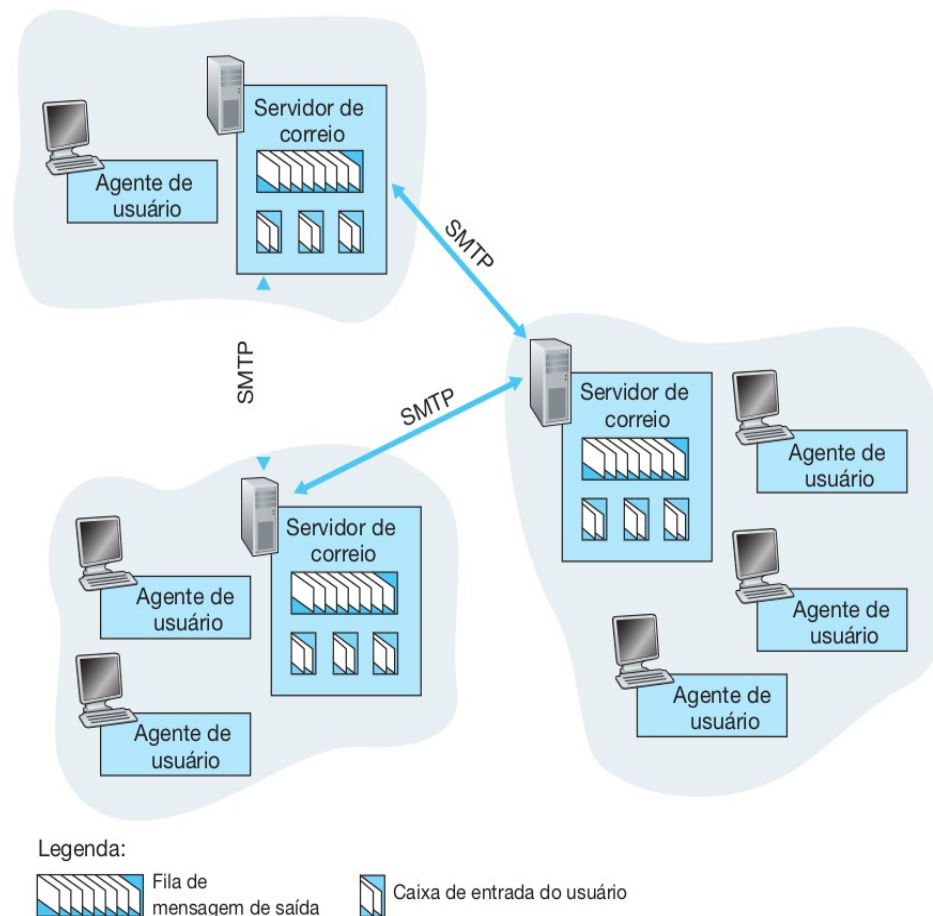
- caixa de correio contém mensagens de chegada (ainda não lidas) p/ usuário
- fila de mensagens contém mensagens de saída (a serem enviadas)
- protocolo SMTP entre servidores de correio para transferir mensagens de correio
 - cliente: servidor de correio que envia
 - “servidor”: servidor de correio que recebe



Correio eletrônico

Servidores de correio

- caixa de correio contém mensagens de chegada (ainda não lidas) p/ usuário
- fila de mensagens contém mensagens de saída (a serem enviadas)
- protocolo SMTP entre servidores de correio para transferir mensagens de correio
 - cliente: servidor de correio que envia
 - “servidor”: servidor de correio que recebe

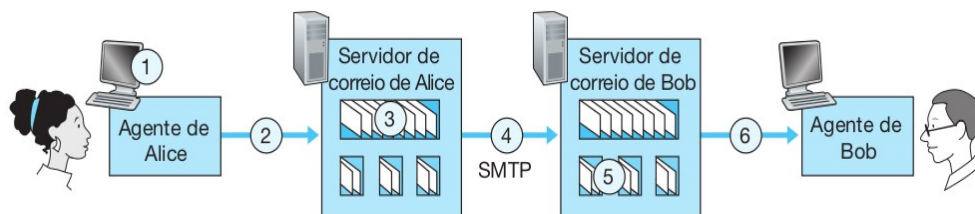


Correio eletrônico: SMTP

- usa TCP para a transferência confiável de msgs do correio do cliente ao servidor, porta 25
- transferência direta: servidor remetente ao servidor receptor
- três fases da transferência
 - handshaking (saudação)
 - transferência das mensagens
 - encerramento
- interação comando/resposta (como o HTTP e o FTP)
 - comandos: texto ASCII
 - resposta: código e frase de status
- mensagens precisam ser em ASCII de 7-bits

Exemplo: Alice envia uma mensagem para Bob

- 1) Alice usa o AU para compor uma mensagem “para” bob@some school.edu
- 2) O AU de Alice envia a mensagem para o seu servidor de correio; a mensagem é colocada na fila de mensagens
- 3) O lado cliente do SMTP abre uma conexão TCP com o servidor de correio de Bob
- 4) O cliente SMTP envia a mensagem de Alice através da conexão TCP
- 5) O servidor de correio de Bob coloca a mensagem na caixa de entrada de Bob
- 6) Bob chama o seu AU para ler a mensagem



Legenda:



Fila de mensagem



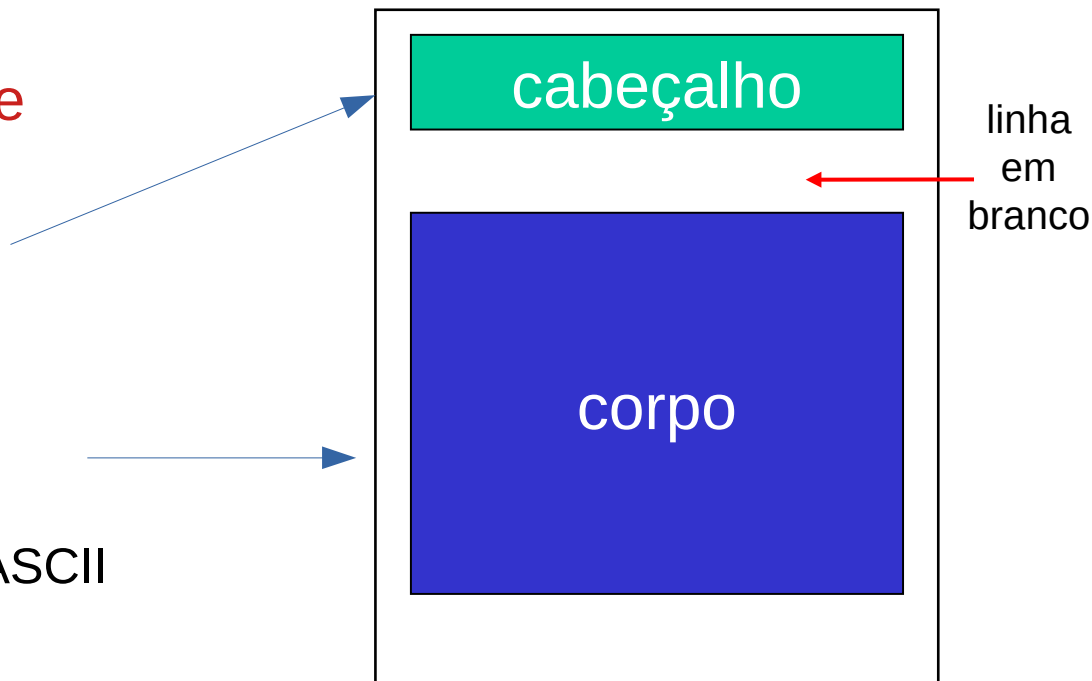
Caixa postal do usuário

SMTP: formato da mensagem

SMTP: protocolo para trocar msgs de correio

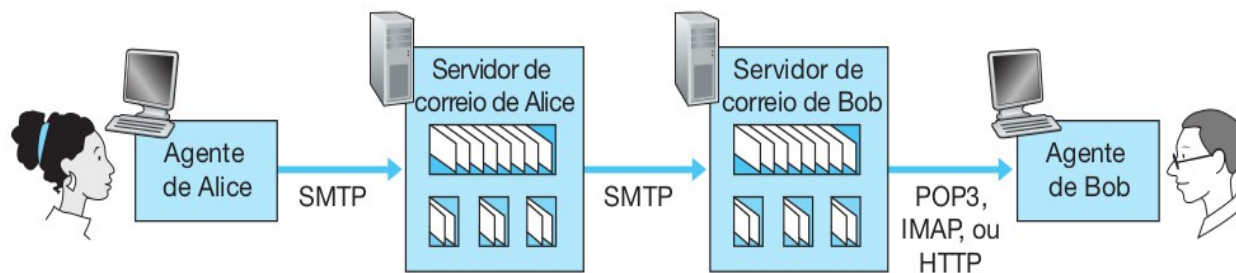
RFC 822: padrão para formato de mensagem de texto:

- linhas de cabeçalho, p.ex.,
 - To, From, Subject....
- corpo
 - A “mensagem” somente em ASCII



Protocolos de acesso ao correio

- SMTP: entrega/armazenamento no servidor do receptor
- protocolo de acesso ao correio: recupera do servidor
- **POP**: Post Office Protocol [RFC 1939]
 - autorização (agente <-->servidor) e transferência
- **IMAP**: Internet Mail Access Protocol [RFC 1730]
 - mais comandos (mais complexo)
 - manuseio de msgs armazenadas no servidor
- **HTTP**: gmail, Hotmail , Yahoo! Mail. etc.



Serviço de nomes

- Pessoas: muitos identificadores:
 - CPF, nome, no. da Identidade
- Hospedeiros e roteadores na Internet :
 - endereço IP (32 bit) - usado p/ endereçar datagramas
 - “nome”, ex., www.yahoo.com – usado por gente
- **Pergunta:** como mapear entre nome e endereço IP?

DSN: Domain Name System

- **Base de dados distribuída**
 - implementada na hierarquia de muitos servidores de nomes
- **Protocolo de camada de aplicação** permite que hospedeiros, roteadores, servidores de nomes se comuniquem para resolver nomes (tradução endereço/nome)
 - nota: função imprescindível da Internet implementada como protocolo de camada de aplicação
 - complexidade na borda da rede

DSN: Domain Name System

Serviços DNS

- Tradução de nome de hospedeiro para IP
- Apelidos para hospedeiros (aliasing)
 - Nomes canônicos e apelidos
- Apelidos para servidores de e-mail
- Distribuição de carga
 - Servidores Web replicados:
 - conjunto de endereços IP
 - para um mesmo nome

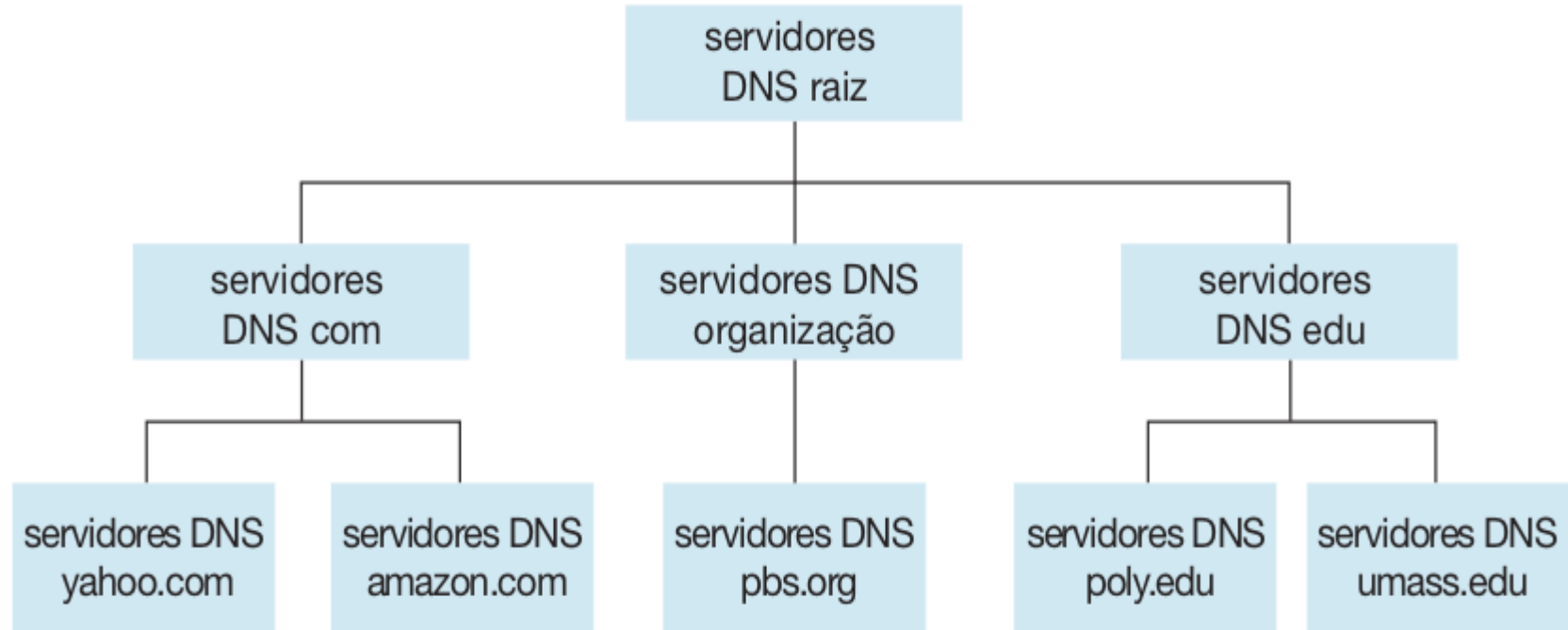
DSN: Domain Name System

- Por que não **centralizar** o DNS?
- Questões sobre **vantagem e desvantagem** da utilização de redes de computadores!

DSN: Domain Name System

- Por que não **centralizar** o DNS?
 - Ponto único de falha
 - Volume de tráfego
 - Base de dados centralizada e distante
 - Manutenção da base de dados
- Não é escalável

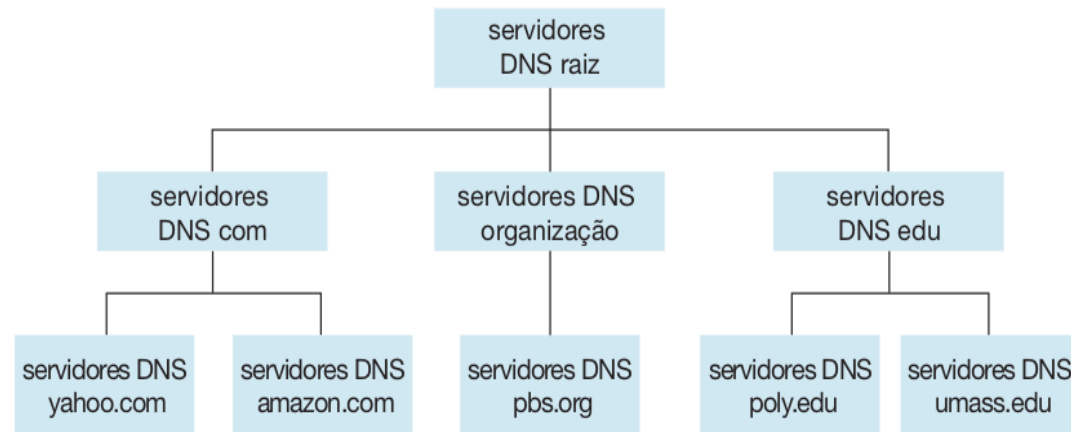
Base de dados hierárquica e distribuída



Base de dados hierárquica e distribuída

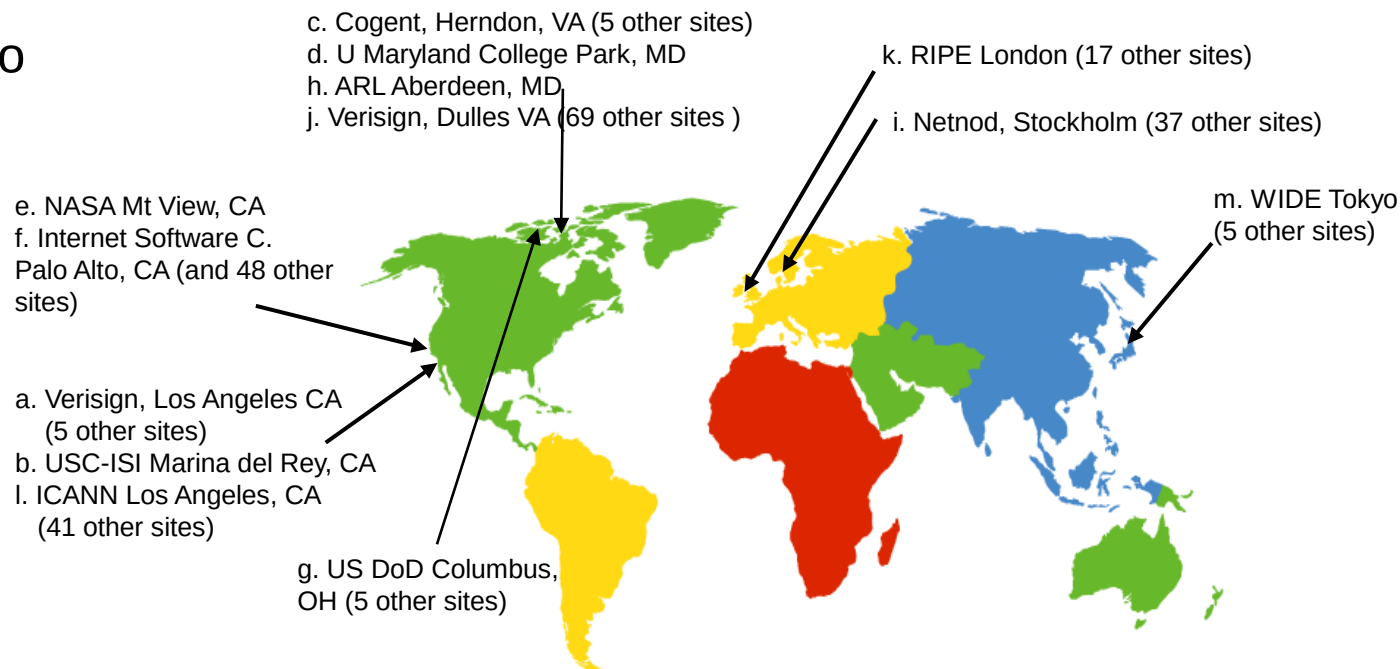
Cliente quer IP para www.amazon.com:

- Cliente consulta um servidor raiz para encontrar um servidor DNS .com
- Cliente consulta servidor DNS .com para obter o servidor DNS para o domínio amazon.com
- Cliente consulta servidor DNS do domínio amazon.com para obter endereço IP de www.amazon.com



DNS: servidor raiz

- procurado por servidor local que não consegue resolver o nome
- servidor raiz:
 - procura servidor oficial se mapeamento desconhecido
 - obtém tradução
 - devolve mapeamento ao servidor local



Servidores TLD e oficiais

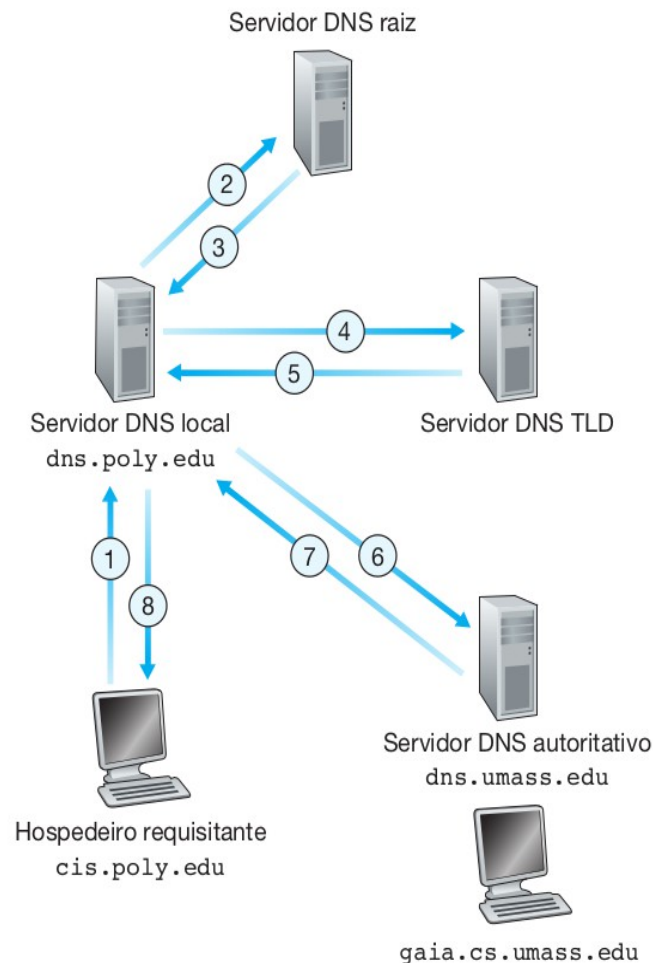
- **Servidores de nomes de Domínio de Alto Nível (TLD):**
 - servidores DNS responsáveis por domínios com, org, net, edu, etc, e todos os domínios de países como br, uk, fr, ca, jp.
 - Domínios genéricos: book, globo, rio
 - Lista completa em: <https://www.iana.org/domains/root/db>
 - NIC.br (Registro .br) para domínio .br (<https://registro.br/>)
- **Servidores de nomes com autoridade:**
 - servidores DNS das organizações, provendo mapeamentos oficiais entre nomes de hospedeiros e endereços IP para os servidores da organização (e.x., Web e correio).
 - Podem ser mantidos pelas organizações ou pelo provedor de acesso

Servidor de DNS local

- Não pertence necessariamente à hierarquia
- Cada ISP (ISP residencial, companhia, universidade) possui um
 - Também chamada do “servidor de nomes default”
- Quanto um hospedeiro faz uma consulta DNS, a mesma é enviada para o seu servidor DNS local
 - Possui uma cache local com pares de tradução nome/endereço recentes (mas podem estar desatualizados!)
 - Atua como um intermediário, enviando consultas para a hierarquia.

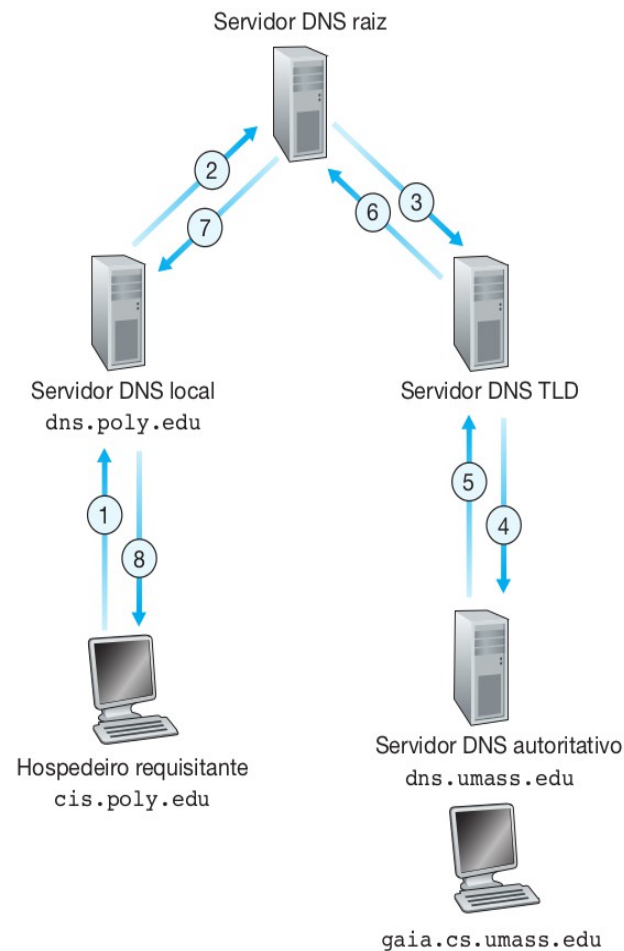
Exemplo de resolução de nome pelo DNS

- Hospedeiro em cis.poly.edu quer endereço IP para gaia.cs.umass.edu
- consulta interativa:
 - servidor consultado responde com o nome de um servidor de contato
 - “Não conheço este nome, mas pergunte para esse servidor”



Exemplo de resolução de nome pelo DNS

- consulta recursiva:
 - transfere a responsabilidade de resolução do nome para o servidor de nomes contatado
 - Carga pesada?



DNS: uso de cache, atualização de dados

- Uma vez que um servidor qualquer aprende um mapeamento, ele o coloca em uma **cache** local
 - entradas na cache são sujeitas a temporização (desaparecem) depois de um certo tempo (TTL)
- Entradas na cache podem estar **desatualizadas** (tradução nome/endereço do tipo melhor esforço!)
 - Se o endereço IP de um nome de host for alterado, pode não ser conhecido em toda a Internet até que todos os TTLs expirem
- Mecanismos de atualização/notificação propostos na RFC 2136

DNS: protocolo e mensagens

protocolo DNS: mensagens de **pedido** e **resposta**, ambas com o mesmo **formato de mensagem**

- cabeçalho de msg
 - identificação: ID de 16 bit para p
 - resposta ao pedido usa mesmo I
- flags:
 - pedido ou resposta
 - recursão desejada
 - recursão permitida
 - resposta é oficial

Identificação	Flags	12 bytes
Número de perguntas	Número de RRs de resposta	
Número de RRs autoritativos	Número de RRs adicionais	
Perguntas (número variável de perguntas)		Nome, campos de tipo para uma consulta
Respostas (número variável de registros de recursos)		RRs de resposta à consulta
Autoridade (número variável de registros de recursos)		Registros para servidores com autoridade
Informação adicional (número variável de registros de recursos)		Informação adicional 'útil', que pode ser usada

Ataque ao DNS

Ataques DDoS

- Bombardeia os servidores raiz com tráfego
 - Até o momento não tiveram sucesso
 - Filtragem do tráfego
 - Servidores DNS locais cacheiam os IPs dos servidores TLD, permitindo que os servidores raízes não sejam consultados
- Bombardeio aos servidores TLD
 - Potencialmente mais perigoso

Ataques de redirecionamento

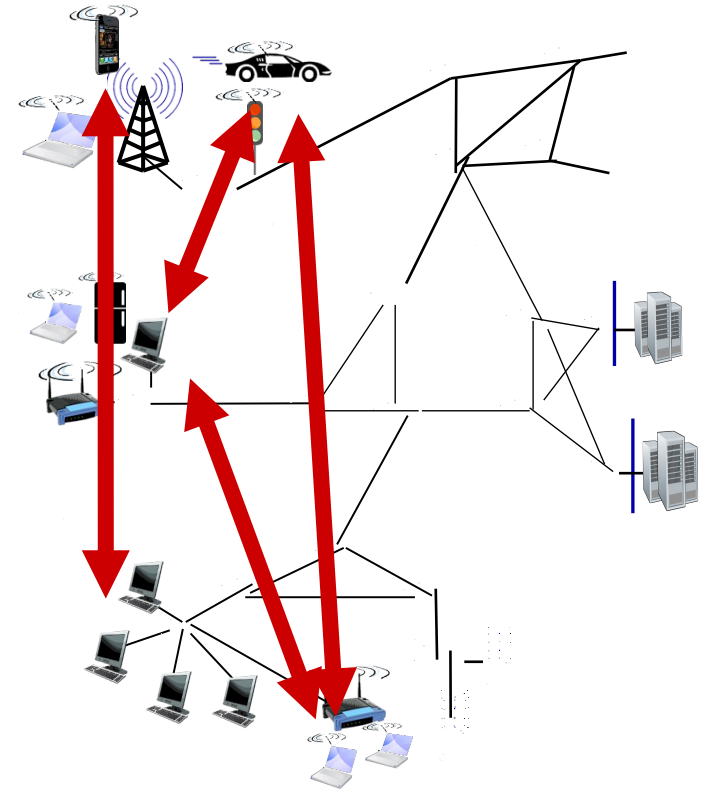
- Pessoa no meio
 - Intercepta as consultas
- Envenenamento do DNS
 - Envia respostas falsas para o servidor DNS que as coloca em cache

Exploração do DNS para DDoS

- Envia consultas com endereço origem falsificado: IP alvo

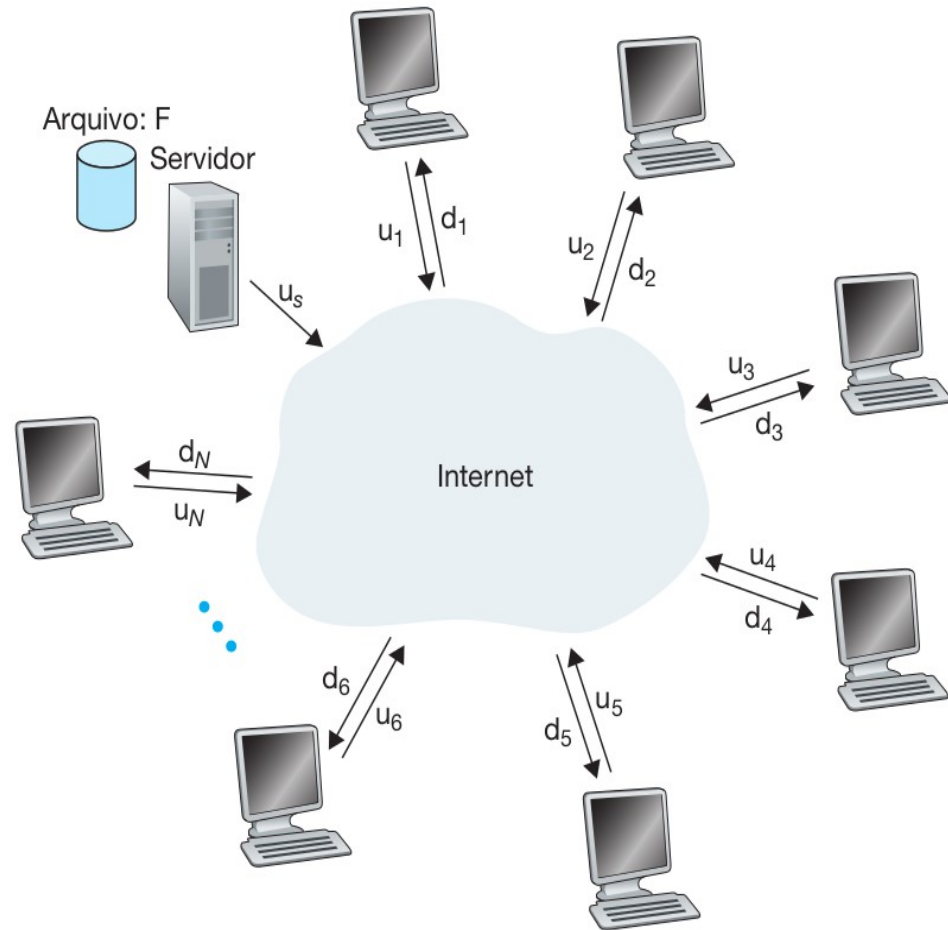
Arquitetura P2P (peer-to-peer)

- Sem servidor sempre ligado
- Sistemas finais arbitrários se comunicam diretamente
- pares estão conectados de forma intermitente e mudam seus endereços IP
 - Escalabilidade: novos peers entram e aumentam a capacidade do serviço/demanda
- Exemplos: Distribuição de arquivos (BitTorrent)
Streaming (KanKan) VoIP (Skype)

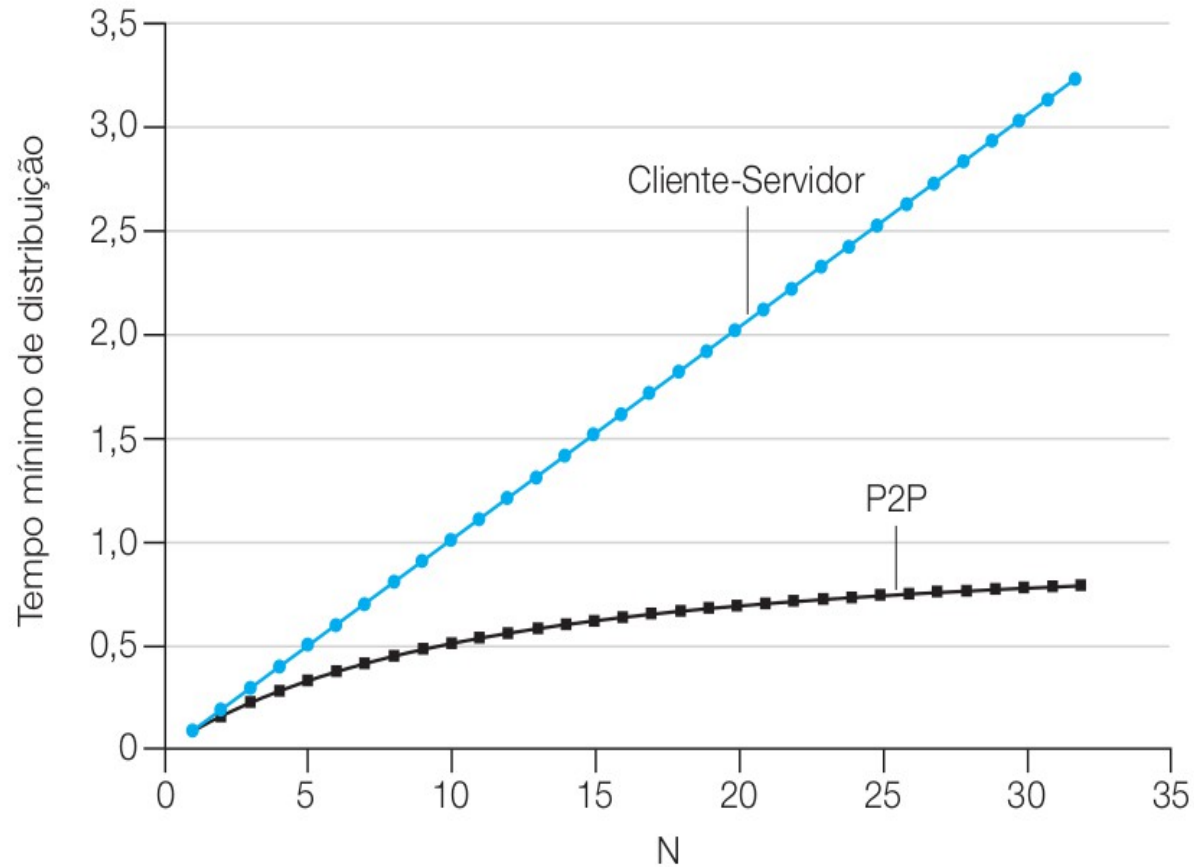


Distribuição de arquivos

Cada computador possui
a sua taxa de
download (d) e
upload (u)

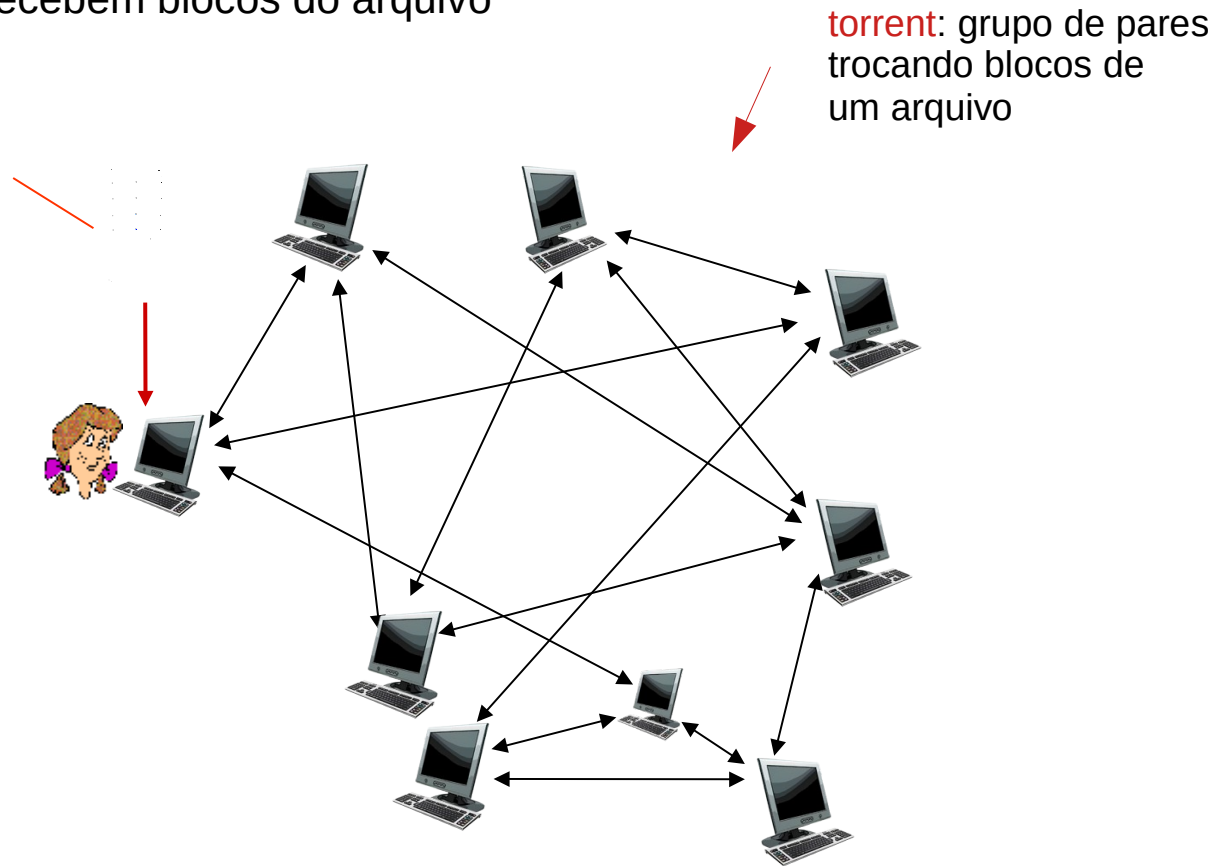


Tempo de distribuição



Distribuição de arquivos P2P: BitTorrent

- Arquivos divididos em blocos de 256kb
- Pares numa torrente enviam/recebem blocos do arquivo



Distribuição de arquivos P2P: BitTorrent

- par que se une a rede:
 - não tem nenhum bloco, mas irá acumulá-los com o tempo
 - registra com o tracker para obter lista dos pares, conecta a um subconjunto de pares (“vizinhos”)
- enquanto faz o download, par carrega blocos para outros pares
- par pode mudar os parceiros com os quais troca os blocos
- pares podem entrar e sair
- quando o par obtiver todo o arquivo, ele pode (egoisticamente) sair ou permanecer (altruisticamente) na rede